# Scratchis toetatavad programmeerimise kontseptsioonid ja oskused

Interaktiivsete lugude, mängude ja animatsioonide koostamise protsessis Scratchiga saab algaja teha endale selgeks mitmed rakenduste loomise ja programmeerimise tähtsad oskused ja kontseptsioonid.

## Probleemide lahendamise ja projektide disaini oskused

* loogiline ja algoritmiline mõtlemine
* süsteemne lähenemine probleemide lahendamisele
* ideede arendus alates lähtekontseptsioonist kuni projekti lõpplahenduseni
* kasutajaliideste loomise oskused ja kogemused
* silumise ja testimise vilumused
* keskendumisvõime ja visaduse arendamine

# Fundamentaalsed ideed arvutitest ja programmeerimisest

* programm ütleb arvutile täpselt, mida teha, samm-sammult
* programmide koostamine ei nõua spetsiaalseid teadmisi, vaid selget ja hoolikat mõtlemist

## Rakenduste loomise ja programmeerimise põhikontseptsioonid ja põhimõisted

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Kontsept | Selgitus | | Näide | |
| **Kasutajaliides** | Sõltumata programmeerimissüsteemist ja ‑keelest, kuulub rakenduse loomise protsessi ka kasutajaliidese disain ja realiseerimine.  K**asutajaliides** sisaldab vahendeid, mille abil saab kasutaja suhelda prog­ram­miga: anda vajalikke korraldusi ja näha nende täitmise tulemusi, muuta algandmeid jm.  Scratch’is luuakse liides **laval**. Selle ele­mentideks võivad olla taustad, aktiivsed ja passiivsed sprai­did, käsunupud, klahvid, muutujate monitorid jm.  Teistes süsteemides kasutatakse selleks tavaliselt vorme ja dialoogibokse. | | [Näita](http://scratch.ttu.ee/Scratch/Opik/Jalka_K.html) | |
| **Programm.**  **Laused** ja **plokid.**  **Programmi­üksused:**  **protseduurid** ja **skriptid** | Programm on **lausete** (korralduste, käsku­de) kogum, mis määrab, milliseid **tegevusi** peab arvuti täitma **andmete** ja/või **objek­ti­dega**, ning tagabka **kasutajaliides**etöö.  Igas keeles on piiratud valik lauseid, mille esitamiseks on kind­lad reeglid.  Scratchis kujutavad **käsud** (laused) endast graafilisi **plokk**e, mis on jagatud otstarbe alusel gruppidesse: **Liikumine**, **Juhtimine** jne. Plokiga on lause süntaks määratletud üheselt ja süntaksi­vigade tekkimine on peaaegu võimatu.  Programm võib koosneda mitmest üksu­sest. Scratch’is nimetakse programmi­üksusi **skriptideks**.  Iga skript on seotud ühe kindla spraidiga ja määrab selle tegevusi. Ühel spraidil võib olla mitu skripti. Skript saab käivitada (öeldakse ka pöörduda, kutsuda välja) teisi skripte, mis kuuluvad samale spraidile või teistele spraitidele. Pöör­du­miseks kasutakse plokke **teavita *nimi*** või **teavita *nimi* ja oota**. | | [Näita](http://scratch.ttu.ee/Scratch/Opik/Proge_Skeem.html)    Programm koosneb neljast skriptist. Kaks on spraidi Kraps omad, Jukul ja Plikal on üks skript.  Kui klõpsatakse rohelist lippu, algab Krapsu esimese skripti täitmine. Peale ”teretamist” käivitab antud skript käsuga **teavita *start* ja oota** korraga Juku ja Plika skriptid, mis algavad plokkidega **kui saabub teade *start***, ning ootab, kuni nende täitmine jõuab lõpuni. Seejärel jätkub Krapsu skripti täitmine. Kraps liigub lava keskpunkti ja käivitatakse sama spraidi teine skript. Peale selle töö lõppu jätkatakse esimese skripti täitmist, mis viib Krapsu etteantud punkti. | |
| **Objektid (spraidid). Objektide**  **omadused, meetodid ja sündmused** | Scratch’is on kesksel kohal **graafilised** **objektid**, mida nimetakse **spraitideks** (*sprite*) ning nende kostüümid. Objekti­deks on samuti ka **lava** ja selle taustad. Kuigi Scratch ei ole formaalselt objekt­orienteeritud süsteem, on sageli otstarbe­kas seda käsitleda tuginedes objekt­orien­teeritud lähenemisviisile.  Iga objektiga on seotud teatud valik **omadusi:**  nimi, asukoht laval (x‑y), suurus, värvus jm, ning **meetodeid**, mille abil määratakse tegevusi antud tüüpi objektiga: asukoha muutmine, pöörami­ne, suuruse ja värvuse muutmine jm. Käsuplokid vastavad sisuliselt meetoditele.  Objekt võib reageerida **sündmustele**: hiireklõps, vajutus etteantud klahvile, kokkupuude teise objektiga. | | Skripti täitmine algab, kui klõpsatakse spraiti Kraps (reaktsioon sündmusele). [Näita](http://scratch.ttu.ee/Scratch/Opik/Objektid.html)  Objekt_1_skript  Skripti plokid muudavad objekti (spraidi) omadusi: suund, asukoht, suurus, värvus, keere. | |
| **Andmete liigid** | Scratchis saab kasutada märk-, graa­fika- ja heliandmeid.  **Märkandmed**: arvud ja tekstid ehk stringid. Neid saab kasutada paljudes plokkides (käskudes) konstantidena, muutujatena ja loendite elementidena. Väärtusi saab leida (tuletada) avaldiste ja funktsioonide abil.  **Graafikaandmed** võivad esineda rakendustes kahes variandis:  **Spraidid** ja lava **taustad** – impordi­tak­se või tehakse graafikaredaktori abil. Käskude abil saab määrata erinevaid tegevusi nendega.  **Pliiatsi** käskudega tehtavad **joonised**  **Heliandmed**. Saab kasutada erinevaid vahendeid helide tekitamiseks (plokid **mängi nooti...**, **mängi trummi...** jm), importida ja lindistada heliklippe: kõne, muusika jm, | | Andmed_L_Skript_1  Skriptid demonstreerivad erinevat liiki andmete kasutamist. Märkandmed (arvud ja stringid) on kasutusel konstantidena ja muutujatena. [Näita](http://scratch.ttu.ee/Scratch/Opik/Andmeliigid.html)  Andmed_L_Skript_2 cat1-a  Graafikaandmeid esindab sprait Kraps ja pliiatsiga tehtav joon spraidi liikumisel.  Heliandmetega on tegemist plokkide **mängi heli** ... ja **mängi trummi**... kasutamisel. | |
| **Andmete organisatsioon** | Organisatsiooni järgi eristatakse kõikides programmeerimiskeeltes:   * **skalaarandmed**: konstandid ja muutujad * **struktuurandmed**: tabelid, massiivid, loendid jm   **Konstantide** väärtused esitatakse vahetult programmis. Ülaltoodud skripti näites on mitu arv- ja stringkonstanti. Need on plokkide väljades asuvad konkreet­sed arvud: 2, 100, 58, 0.2, 0 jne; ja tekstid: ”Tere! Mina olen Kraps!”, ”Mis on Sinu nimi?” jne.  **Muutujad** esitatakse programmis nimede abil, nende väärtusi programmis (skriptis) ei näe. Väärtused tekkivad tavaliselt programmi täitmise ajal. Näiteks muutuja *pikkus* väärtus tekkib sisestamisel käsuga **küsi**, muutuja *normkaal* väärtus tekkib arvutuste tulemusena  (*pikkus* – 100). Muutujate olemusest ja käsitlemisest vt allpool.  Tabelid, massiivid, **loendid** kujutavad endast kindla struktuuriga väärtuste kogumeid. Scratch’is saab kasutada ainult **loendeid**, mis on massiivide erijuht (vt. allpool) | | | |
| **Muutujad ja omistamine** | Muutuja on **mälu­pesa**, kuhu programm saab salvestada väärtusi: arve ja stringe ning kasutada (lugeda) neid hiljem näiteks uute väärtuste leidmiseks.  Scratch’is saab muutujaid luua ja kasuta­da grupis **Muutujad** olevate plokkide abil. Loomisel saab anda muutujale **nime**, mida kasutatakse käskudes viitamiseks tema jooksvale väärtusele. Loomisel saab ka määrata, kas muutuja on kättesaadav kõikidele spraitidele (globaalne muutuja) või ainult ühele konkreetsele spraidile (lokaalne muutuja).  Muutujatele väärtuste omistamiseks ja väärtuste muutmiseks kasutatakse plokke  **võta** *muutuja* = *avaldis* ja  **muuda** *muutuja* *avaldis* võrra  Muutuja väärtust saab kuvada laval nn **monitori** abil 0_monitor. Väärtust saab muuta ka ”käsitsi” **liuguri**ga. | Programm imiteerib korvpalliviskeid  **Korvpall_Skript_2**  **Näited:** [**Korvpall**](http://scratch.ttu.ee/Scratch/Opik/Korvpall.html) **ja** [Ideaal](http://scratch.ttu.ee/Scratch/Opik/Ideaal.html) | | |
| **Loendid (massiivid)** | Scratch’i loend vastab enam-vähem teis­tes keeltes kasutatavatele dünaamilistele ühe­mõõtmelistele massiividele. Taoline massiiv kujutab endast järjestatud mälu­pe­sade (elementide) kogumit. Elementide­le saab viidata massiivi nime ja indeksite abil.  Scratch’is saab luua ja kasutada loendeid grupi **Muutujad** plokkidega. Viimaste abil saab lisada elemente loendi lõppu ja vahele, asendada ja eemaldada elemente, viidata elementidele jm. | Loendid_Monitorid  Loendid_Arvutid_Hinnad  Skript keskmise hinna leidmiseks  Loendid_Kesk_Skript  [Näita](http://scratch.ttu.ee/Scratch/Opik/Loendid.html) | | |
| **Avaldised, tehted** ja **funktsioonid** | Avaldise abil saab anda eeskirja vajaliku väärtuse leidmiseks. Avaldiste loomiseks kasutatakse Scratch’is grupi **Tehted** plokke.  Avaldis koosneb operandidest ja tehetest.  Operandideks võivad olla: konstandid, muutujad, funktsioonid, loendite elemendid.  Sõltuvalt tehetest võib avaldised jagada järgmistesse rühmadesse:   * arvavaldised: +, -, \*, / * stringavaldised: ühenda, eralda * võrdlused: <, =, > * loogikaavaldised: **ja**, **või, mitte**   Avaldistes saab kasutada mitmeid funk­tsioone: **abs**, **sqrt** (ruutjuur), **sin**, **cos**, **asin**, **log**, **ln** jm.  Sisuliselt on funktsioonideks ka eraldi esitaud plokid; **mod** (jääk), **ümarda** ja **juhuarv.**  **NB!** Sulge tehete järjekorra määrami­seks Scratch’is kasutada ei saa. Iga tehte plokki võib lugeda sulgudes olevaks avaldise osaks. | Näide [Ideaal](http://scratch.ttu.ee/Scratch/Opik/Ideaal.html) | | |
| **Joonistamine,**  **joonestamine** | Joonis_piltEnamikes programmeerimiskeeltes on olemas vahendid, mis võimaldavad programmil luua jooniseid.  Scratch’is on peamised vahendid selleks gruppides **Pliiats** ja **Liikumine**.  Kõrvalolev skript joonistab täisnurkse kolmnurga, mille külgede pikkused **a** ja **b** tekitatakse juhu­arvude abil.  Täisnurk pannakse lava keskele: punkti (0,0). Muutuja **m** abil määratakse mas­taap: punktide arv ühes pikkusühikus. | Demo [Rist\_Ring](http://scratch.ttu.ee/Scratch/Opik/Joonestamine.html) | | |
| **Märkandmete sisestamine ja väljastamine** | Ülesannete lahendamisel on prog­ram­mil sageli vaja saada algandmeid ja peege­lda­da tulemusi. Esimesel juhul räägitakse andmete **sisestamisest** (lugemisest) , teisel juhul andmete **väljastamisest** (kirjutami­ne, kuvamine).  Scratch’is saab andmete sisestamiseks kasutada muutujate liuguritega monitore (ainult arvud) ja plokki **küsi**, mis võimal­dab sisestada arve ja tekste dialoogi­režiimis.  Tulemusi saab väljastada muutujate ja/või loendite **monitoridega** või plokkidega **ütle**. | Näide [Ideaal](http://scratch.ttu.ee/Scratch/Opik/Ideaal.html) | | |
| **Protsesside juhtimine** | Väga tähtsal kohal ülesannete lahendamisel on tegevuste täitmise järjekorra määramine. Just selles seisneb programmeerimise kunst. Vajalikud tegevused ja nende järjekord ei sõltu üldiselt kasutatavast keelest, vaid tulenevad ülesande olemusest ehk selle lahendamise **algoritmist**.  Võib eristada nelja liiki protsesse:   * **jada** ehk järjestikune protsess, * **kordus** ehk tsükliline protsess, * **valik** ehk hargnev protsess ning * **paralleelne protsess**.   Programmeerimiskeeltes on olemas vahendid erinevat liiki protsesside kirjeldamiseks. | | | |
| **Jada** ehk järjestikune protsess | Programmi koostamisel peab arvestama, et plokkide poolt määratud tegevusi täide­takse kindlas järjekorras.  Liht­samal juhul on tegemist järjestikuse protsessiga, kus plokke täidetakse järjest ülevalt alla. | | | Jada_pilt |
| **Kordus** ehk tsükliline protsess | Korduste kirjeldamiseks on Scratch’is, nagu ka teistes keeltes, mitu lauset (plokki): **lõputult**, **korda** ***n***, **korda kuni *tingimus***, **lõputult kui *tingimus***.  Ploki **lõputult** sees olevaid plokke täidetakse põhimõtteliselt lõpmatult. Skripti töö saab katkestada näiteks punase nupuga. Korduse sees võib olla üks või mitu plokki **kui *tingimus***, mis tingimuse täitumisel katkestab vastavate plokkide abil skripti (plokk **peata skript**) või terve programmi töö (**peata kõik**). | | | Esimese ploki toimel ”jalutab” sprait lõpma­tult laval edasi-tagasi. Katkestada saab punase nupu abil. Teise ploki täitmisel liigub sprait vasakult paremale. Liikumine (kordamine) lõpeb, kui spraidi x-koordinaat saab suuremaks 200-st. |
| Käsu **korda *n*** täitmisel korratakse ploki sees olevaid plokke ***n*** korda. Kordamiste arv (***n***) võib olla antud konstandi, muutuja või avaldise abil. | | | korda_n_korda  Objekt (näiteks kiisu) ”põristab” trummi ja teeb ringi (liikudes ja pööreldes 360o) laval [Näita](http://scratch.ttu.ee/Scratch/Opik/Kordused_1.html) |
| Käsu **korda kuni *tingimus*** täitmisel korratakse ploki sees olevaid plokke seni, kuni tingimus saab tõeseks. | | | kordus_kuni |
| Käsu **korda kui *tingimus*** täitmisel korratakse ploki sees olevaid plokke seni, kui tingimus on tõene | | | kordus_kui |
| **Valik** ehk hargnev protsess | Plokkidega **kui *tingimus*** ja **kui *tingimus* muidu** saab kirjeldada valikuid ehk hargnevaid protsesse.  Lause **kui ...** korral (valik ühest) täide­takse ploki sees olevad plokid, kui ***tingimus*** on tõene, vastupidisel juhul jäetakse need vahele.  Lause **kui** **... muidu** korral (kahendvalik). Kui ***tingimus*** on tõene, täidetakse esimeses harus olevad plokid, vastupidisel juhul teises harus olevad plokid. | | | Esimeses plokis kontrollitakse tingumust, kas  x-koordinaat on suurem 200-st, kui jah, täidetakse sisemised plokid. Teises skripti fragmendis kuvatakse teade sõltuvalt sisestatud vanusest. Näide [Ideaal](http://scratch.ttu.ee/Scratch/Opik/Ideaal.html) |
| **Harud** ehk paralleelsed protsessid | Kahte või enamat skripti (protsessi) saab täita samaaegselt ehk paralleelselt.  Paralleelset täitmist saab määrata mitmel erineval viisil.  Näiteks kõik skriptid, mille esimeseks plokiks on rohelise lipuga plokk, käivitatakse samaaegselt ja täidetakse paralleelselt, kui klõpsatakse rohelist lippu.  Paralleelselt täidetakse ka skripte, mis algavad plokiga **kui saabub teade** ***nimi***, milles on kasutusel sama ***nimi***. | | | Üks objekt liigub pidevalt mööda kolmnurkset trajektoori, teine mööda ringikujulist.    Kui saabub teade ”korras”, teeb üks objekt 3 hüpet, teine vahetab 8 korda kostüümi. Vt. projekt [Tantsud](http://scratch.ttu.ee/Scratch/Juhend/Tantsud.html) |
| **Sündmused** | Objektid võivad reageerida teatud sünd­mus­tele: vajutus mingile klahvile, objekti klõpsamine hiirega, kokkupuude teise objektiga või lava servaga jm. Skriptides võib ette näha reaktsioone kindlatele sündmustele. Peamisteks vahenditeks sündmuste haldamisele on nn päiseplokid:  **kui vajutatakse klahvi** ja  **kui klõpsatakse spraiti**  Skriptide sees leiab sageli kasutust plokk  **kui puudutab** {sprait | serv | kursor}  Kui antud sprait puudutab teist spraiti, lava serva või hiire kursorit, tagastab plokk väärtuse ***tõene***.  Põhimõtteliselt on sündmusega tegemist ka rohelise lipu kasutamisel ja skriptide käivitamisel plokkide **teavita** ja **kui saabub teade** abil (vt allpool). | | | Kui klõpsatakse rohelist lippu, võetakse muutujate väärtuseks 0 ja pall läheb algseisu  Kui vajutatakse tühikut, muudetakse juhuarvudega palli asukohta ja suurendatakse löökide arvu ühe võrra. Kui pall puudutab väravat, lisatakse 1 muutujale tabas. Proovige! |
| **Dünaamiline interaktsioon** | Objekte saab mõjutada nö reaalajas hiire kursoriga (hiire liigutamisega), heliga jm., kasutades grupi **Andurid** plokke **hiire *x***, **hiire *y***, **helitugevus** jm | | | Kui klõpsatakse lippu, liigub objekt horisontaal­suunas koos hiire kursoriga ja muudab omaduse keere suurust, vastavalt y väärtusele.  KeereProovige! |
| **Koostöö sprai­tide ja skrip­tide vahel**. | Kui programm koosneb mitmest skriptist tekib sageli vajadus nende töö koordi­nee­rimiseks ja sünkroniseerimiseks.  Üks skript võib pöörduda teiste poole, need omakorda järgmiste poole jne.  Skrip­tide koostöö korraldamisel kasutatakse plokke  **teavita** *teade* ja  **teavita** *teade* **ja oota** ning  **kui saabub** *teade*  Siin vaadeldav programm koosneb neljast skriptist. Rohelise lipuga skript on **peaskript**. Sellest algab programmi töö ning ta käivitab teisi. Kui kasutaja sisestab tähe ”d”, käivitatakse skript **Loe** ning peale selle töö lõppu pannakse tööle skript **Arvuta**. Kui sisestati d-st erinev väärtus, käivitatakse kohe **Arvuta**. | | | Ootamata **Arvuta** töö lõppu, pannakse tööle ka skript **Joonesta** (ei ole siin näidatud, on analoogne skriptiga punktis J**oonestamine**). [Näide](http://scratch.ttu.ee/Scratch/Opik/Proge_Skeem.html). |
| **Algoritmimine** | Algoritmi all mõistetakse eeskirja, mis määrab, milliseid tegevusi ja mis järjekorras peab täitma antud ülesande lahendamiseks või töö täitmiseks. Vajalikud tegevused ja nende täitmise järjekord üldiselt ei sõltu konkreetsest keelest vaid ülesandest või mudelist. Tegevused algoritmis esitatakse üldisemal kujul. Viimasel ajal kasutatakse algorit­mide esitamiseks sageli modelleerimis­keelt UML.  Scratch’i skripte võib vaadelda kui ühte algoritmide esitusviisi. | | | Algo_Kraps_Skript  Demod: [Jalka](http://scratch.ttu.ee/Scratch/Juhend/Jalka.html), [Ruutvõrrand](http://scratch.ttu.ee/Scratch/Juhend/Ruutvorrand_0.html), [Maks\_Vek](http://scratch.ttu.ee/Scratch/Juhend/Maks_Vektoris.html) |
| **Modelleeri­mine** | Scratch’i vahendeid saab kasutada objekt­orienteeritud modelleerimise põhimõistete selgitamiseks ja illustreerimiseks.  Scratch’i spraidid kuuluvad ühte universaalsesse **klassi - Sprait**. Spraitidel on kindel valik **omadusi**: nimi, asukoht, suurus , ... ja **meetodeid**: liigu (), pööra (), muudaX(), ... Viimased esitakse käskude ehk plokkide abil.  Käskudest (baasmeetoditest) moodusta­tuid skripte võib käsitleda kasutaja **alam­klassidena**. Alamklassidena on käsitleta­vad ka spraidiga seotud kostüümid, pliiats, muutujad, loendid.  **Lava** on vaadeldav nö peaklassina: klass **Sprait** on selle alamklass: kõik spraidid asuvad laval. | | | Mudel_1  Näide. Projekt [Tantsud](http://scratch.ttu.ee/Scratch/Juhend/Tantsud.html). |

|  |
| --- |
| [scratchi põhiobjektid](http://rlpa.ttu.ee/scratch/Objektid_Scratchis.pdf) |

|  |  |
| --- | --- |
|  |  |